VaultLine Transaction Interface Guide

Myriad Technologies, Inc.
PO Box 2193
Carlsbad, CA 92018
Tel (760) 845-9725
email support@vaultline.com

Table of Contents

1. Methods of Submitting a Transaction

1. Method 1 – Virtual Terminal

Using an internet browser (IE 5.x+, FireFox 1.5+, Mozilla, Safari 2.x+) go to https://vaultline.mydtx.net. There you have a choice of 7 kinds of credit card transactions (Sale, Preauth, Preauth Complete, Force, Void, Return and Settle Batch). All transactions can be entered using a keyboard and mouse or magnetic swipe reader (Card Present Transactions). This is a secure site using up to 256 bit encryption depending on which browser you are using.

For detailed instructions for when to use each transaction go to https://vaultline.mydtx.net/gateway.php?page=help after you log in to the virtual terminal

2. Method 2 – Webservices

If you have your own web site or client application and you want to collect credit card information on your side and send VaultLine a secure transaction over the internet, we provide sample code that sends the payment data via a SSL connection.

2. Virtual Terminal

URL: https://vaultline.mydtx.net

Test Account Login = vitale

Password = test007

The login page brings you to Virtual Terminal main page where you can choose one of seven transactions or run one of 3 canned reports.

> **Note:** This is a test account and does not process live transactions. To process real transactions you must us the login and password provided by VaultLine. Also the Vaultline system is considered a true soft terminal which means you will always process transactions in a terminal environment (NOT HOST). In this terminal environment throughout the day you will be building a batch (or several batches) of transactions (these can be authorizations and/or returns) and at a specified time the batch is submitted for processing. Once a batch is settled and cleared, the movement of funds apply.

1. Normal Use

Normally you will run your sales, voids and returns throughout the normal business day; from time to time you might want to view a report of the current open batch(s).

At the end of the day you will want to verify your batch (unsettled). Once all the transactions are ready (i.e. any voids have been completed) you will want to Settle (upload) it.

Upon successful settlement you will see the batch has moved from the unsettled to the history report. Note: Once a transaction is voided from the current batch it will be moved to the history report for logging.

2. Credit Card: Sale

The SALE transaction is one of the most commonly used functions. Its purpose is to authorize a hold of funds on a specific credit card account. Upon a successful SALE the transaction will enter the currently OPEN/UNSETTLED BATCH.

Procedure:
1. Enter all the required and optional information into the active window.
2. Once you have entered the relevant data and have chosen the desired processing method, press/click the 'submit' button..
3. This transaction is now in the current OPEN/UNSETTLED BATCH.

Required Data: There are several data fields that must be present for a sale transaction to be processed successfully. At a minimum, the Amount, Account and Exp(expiration) date must be passed.

Optional Data: Optional data fields fall into two categories.
Reporting: Depending on the auditing requirements of your organization, you might want to send extra transaction data that can be used to filter/match future system reports.

Cardholder Info: The Cardholder tab allows entry of the following data. Address and Zip fields represent Address Verification Service(AVS) data. The CV field represents the Cardholder Security Code and is used for card Not-Present transactions. The Name on Card field will be automatically populated when swiping a present card, and can be manually entered for auditing purposes.

Order Info: The Order tab allows entry of the following data. The Order # represents a field that can be matched for future reports and should be sent for 'Purchase Card Level II" transactions. The Clerk ID and Comments fields are both used for reporting.

Purchase Card Level II: If you wish to qualify for Level II data then you must pass both the Order # and the Tax field.

3. Credit Card: Preauth

The Pre-authorization transaction is another commonly used function in Testvaultline Its main purpose is to authorize a hold of funds on a specific credit card account, but continue to hold the funds in an 'un-captured' state until goods are delivered/shipped. Upon a successful Preauthorization the transaction will be 'authorized' but not enter the currently OPEN/UNSETTLED BATCH. To complete this transaction see the Pre-auth Complete function.
Note: Pre-auth transactions are mainly used in a card-not-present environment.

Procedure:
1. Enter all the required and optional information into the active window.
2. Once you have entered the relevant data press/click the 'submit' button.

Required Data: There are several data fields that must be present for a pre-authorization transaction to be processed successfully. At a minimum, the Amount, Account and Exp(expiration) date must be passed.

Optional Data: Optional data fields fall into two categories.
1) Reporting: Depending on the auditing requirements of your

organization, you might want to send extra transaction data that can be used to filter/match future system reports.

Cardholder Info: The Cardholder tab allows entry of the following data. Address and Zip fields represent Address Verification Service(AVS) data. The CV field represents the Cardholder Security Code and is used for card Not-Present transactions. The Name on Card field will be automatically populated when swiping a present card, and can be manually entered for auditing purposes.

Order Info: The Order tab allows entry of the following data. The Order # represents a field that can be matched for future reports and should be sent for 'Purchase Card Level II" transactions. The Clerk ID and Comments fields are both used for reporting.

Purchase Card Level II: If you wish to qualify for Level II data then you must pass both the Order # and the Tax field.

4. Credit Card: Force

Although not often used, the 'Force' transaction is sometimes a required function to successfully process a transaction. Its main purpose is to afford the merchant a way to insert an 'offline-authorized' transaction into the currently open batch (as a fully captured transaction). An example of a force is when you get a 'CALL PROCESSOR' response message and the helpdesk gives you a verbal approval/authorization number.

Procedure:
1. Enter all the required and optional information into the active window.
2. Once you have entered the relevant data, press/click the 'submit' button.

Required Data: There are several data fields that must be present for a force transaction to be processed successfully. At a minimum, the Amount, Account Exp(expiration) date and Approval Code must be passed.

Optional Data: Optional data fields fall into two categories.
Reporting: Depending on the auditing requirements of your organization, you might want to send extra transaction data that can be used to filter/match future system reports.

Approval: The Approval tab allows entry of the verbal approval code.

Order Info: The Order tab allows entry of the following data. The Order # represents a field that can be matched for future reports and should be sent for 'Purchase Card Level II" transactions. The Clerk ID and

Comments fields are both used for reporting.

Purchase Card Level II: If you wish to qualify for Level II data then you must pass both the Order # and the Tax field.

5. **Credit Card: Return**

   The 'Return' transaction is used to refund the value of a transaction from the merchant to a cardholder account. Note: A return should only be used if the original transaction has already been successfully settled and posted. For an unsettled transaction a 'void' should be used instead.

   Procedure:
   1. Enter all the required and optional information into the active window.
   2. Once you have entered the relevant data, press/click the 'submit' button.

   Required Data: There are several data fields that must be present for a return transaction to be processed successfully. At a minimum, the Amount, Account and Exp(expiration) date must be passed.

   Optional Data: Optional data fields fall into two categories.
   1) Reporting: Depending on the auditing requirements of your organization, you might want to send extra transaction data that can be used to filter/match future system reports.

   Order Info: The Order Info space allows entry of the following data. The Order # represents a field that can be matched for future reports. The Clerk ID and Comments fields are both used for reporting. The Cardholder Name is used for reporting.

6. **Credit Card: Void**

   Procedure:
   The 'Void' transaction is used to remove a transaction from the currently open/pending batch.

   When you select the 'Void' transaction control the active window will populate a table with all the currently pending transactions (each in a row, and for all open batches).

   1. Select all transactions you wish to remove from the open batch by setting the check-box located in the void column of the table.
   2. Once you have selected the proper transaction(s) to void, press/click the 'submit' button.

7. Credit Card: Preauth Complete

The 'Preauth Complete' transaction is used to complete a previously authorized transaction. Upon success the transaction will be captured into the pending/open batch. Note: You can view the unsettled transactions report for details and status verification.

Procedure:
When you select the 'Preauth Complete' transaction control the active window will populate a table with all the currently pending 'preauth' transactions (each in a row, and for all open batches).

1. Select all transactions you wish to complete from the open list by setting the check-box located in the -Comp- column of the table.
2. Once you have selected the proper transaction(s) to complete, press/click the 'submit' button.

8. Administration: Reports

The 'Report' transaction is used to perform systems trace/audit functions from within the webclient. Note: There are several different types of reports from which to choose and each successful report transaction may be saved and/or printed.

Procedure:

1. Select the report type you wish to view. Your choices are 'Unsettled' (all open/pending batch-settlement), 'Settled' (a history of all transactions which have been settled or voided), 'Failed' (all transactions that were not successfully completed, for one reason or the other) and 'Batch Totals' (the current summary for each open batch by card/transaction type).
2. Choose to filter this report by date.
3. Choose to use other filters for this report.
4. Once you have selected the proper report type and have chosen your filters press/click the 'submit' button.

9. Administration: Settle

At the end of the day it is the 'Settle' transaction which is one of the most important functions performed. It is when any/all current batches are cleared off the processing host and when funds are actually cleared for settlement to the merchants bank account. Note: This is when you are actually getting paid.

Procedure:
1. Verify the current batch has the proper information in regards to valid transactions. For instance, you would want to verify all tips and/or adjustments have been made and that any transactions needing removal

have been voided.
2. Choose to 'Upload' the settlement file or to 'Force' it offline.
3. Once you have verified the batch is ready to go and have chosen the proper settlement method press/click the 'submit' button.

10. Appendix

Amount: The Amount field represents the total dollar amount of the transaction. This amount is positive for a sale transaction and negative for a return. Note: The value is absolute and can be entered with or without a decimal character.

Account: The Account field represents the card or issuer number. Note: Either Account number AND Expiration Date OR trackdata must be passed.

Expiration Date: The Exp field represents the Month and Year of Account expiration.

Process Offline: The Process Offline flag represents if the transaction should be added to the current 'pending authorization' set or if it should be processed in real-time. Note: See the [Admin-> 'Offline Complete'] function for further reference.

Tax: The Tax field represents the total amount of tax (in an absolute dollar form)
example: (111 = $111.00 | 1.1 = $1.10 | .23 = $0.23 | 2.56 = $2.56)

Address: The Address field represents the Street portion of the Card Holder Address (as is reflected on a billing statement). Note: It is important to get any numerical values correct (such as 9401 NW 15th).

Zip: The Zip field represents the Zip Code portion of the Card Holder Address (as is reflected on a billing statement). Note: This number is best represented as a five digit number (i.e. 32615).

Name: The Name field represents the Cardholder name as is present on the card and can be used for future reporting.

CV: The CV value, or Cardholder Security Code, represents the three or four digit code on most major cards. This field should be used in a card not present environment to help curb fraud. Note: Due to the dynamics of how CV authentication works, you might get an authorization with a bad CV value from one card issuer and a denied transaction from a bad CV value with another issuer. At the end of the day, the merchant must use his/her best judgment to either void the transaction, or take a chance with the bad CV and 'authorized' transaction.

Order: The Order # field represents the merchant order or invoice number. This number is passed to select processors, and is stored within the logs for future audits.

Clerk ID: The Clerk ID field represents the Clerk who is administering this sale. Note: This parameter is used exclusively for reporting/auditing purposes.

Comments: The Comments field represents an open field for merchants to add business specific data. Note: This field is alpha numeric and can store up to xx characters.

Approval: The Approval Code represents the verbal authorization number associated with the transaction.

3. Webservice Implementations:

1. Webservice URLs:

Test: https://testweb.mydtx.net/CCWS/CCWS.cfc?wsdl
Production: https://webservices.mydtx.net/CCWS/CCWS.cfc?wsdl

2. Webservice Credit Card Transactions

Method:CCSale (Card Not Present)

returnCCWSResponseStruct
Process Credit Card Sale

Parameters:

Parameter	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
CardNumber	yes	string	CardNumber 20 Characters Max.
CardExpirationMonth	yes	numeric	Numeric month (MM). 2 Digits Max.
CardExpirationYear	yes	numeric	Numeric year (YY). 2 Digits Max.
CardHolderName	yes	string	CardHolderName. 50 Characters Max.
AVSAddress	yes	string	AVSAddress
AVSZip	yes	string	AVSZip
CVV2	yes	string	CVV2
OrderNumber	yes	string	OrderNumber
TaxAmount	yes	numeric	TaxAmount
ClerkID	yes	string	clerked for the transaction.
StationID	yes	string	stationed for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCSaleCP (Card Present)

returnCCWSResponseStruct
Process Credit Card Sale Card Present

Parameters:

Parameter	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
TrackData	yes	string	Card Track Data (pass all TrackData together if available i.e. Track 1 and Track 2)
OrderNumber	yes	string	OrderNumber
TaxAmount	yes	numeric	TaxAmount
ClerkID	yes	string	clerked for the transaction.
StationID	yes	string	stationed for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCPreAuthorization (Card Not Present)

returnCCWSResponseStruct
Process Credit Card Pre-Authorization

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
CardNumber	yes	string	CardNumber 20 Characters Max.
CardExpirationMonth	yes	numeric	Numeric month (MM). 2 Digits Max.
CardExpirationYear	yes	numeric	Numeric year (YY). 2 Digits Max.
CardHolderName	yes	string	CardHolderName. 50 Characters Max.
AVSAddress	yes	string	AVSAddress
AVSZip	yes	string	AVSZip
CVV2	yes	string	CVV2
OrderNumber	yes	string	OrderNumber
TaxAmount	yes	numeric	TaxAmount
ClerkID	yes	string	clerkid for the transaction.
StationID	yes	string	stationid for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCPreAuthorizationCP (Card Present)

returnCCWSResponseStruct
Process Credit Card Pre-Authorization Card Present

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
TrackData	yes	string	Card Track Data (pass all TrackData together if available i.e. Track 1 and Track 2)
OrderNumber	yes	string	OrderNumber
TaxAmount	yes	numeric	TaxAmount
ClerkID	yes	string	clerkid for the transaction.
StationID	yes	string	stationid for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCPreAuthCompleteRequest

returnCCWSResponseStruct
Process Credit Card Delayed Capture with tax amount

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
TransactionID	yes	string	TransactionID of successful PREAUTH transaction.
PTrannum	yes	string	PTrannum
Tax	yes	numeric	Tax Amount
ClerkID	yes	string	clerked for the transaction
StationID	yes	string	stationed for the transaction

Method:CCDelayedCapture

returnCCWSResponseStruct
Process Credit Card Delayed Capture **(NOTE: Not Level 2 compliant! Use CCPreAuthCompleteRequest for Level 2 compliant delayed captures)**

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
TransactionID	yes	string	TransactionID of successful PREAUTH transaction.
PTrannum	yes	string	PTrannum

Method:CCForce

returnCCWSResponseStruct
Process Credit Card Force

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
CardNumber	yes	string	CardNumber 20 Characters Max.
CardExpirationMonth	yes	numeric	Numeric month (MM). 2 Digits Max.
CardExpirationYear	yes	numeric	Numeric year (YY). 2 Digits Max.
AuthNum	yes	string	AuthNum from processor (usually by phone on a contact CC company return).
CardHolderName	yes	string	CardHolderName. 50 Characters Max.
OrderNumber	yes	string	OrderNumber
TaxAmount	yes	numeric	TaxAmount
ClerkID	yes	string	clerkid for the transaction.
StationID	yes	string	stationid for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCVoid

returnCCWSResponseStruct
Process Credit Card Void

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
TransactionID	yes	string	TransactionID of successful PREAUTH transaction.
PTrannum	yes	string	PTrannum

Method:CCReturn

returnCCWSResponseStruct
Process Credit Card Return

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Amount	yes	numeric	Amount
CardNumbe	yes	string	CardNumber 20 Characters Max.
CardExpirationMonth	yes	numeric	Numeric month (MM). 2 Digits Max.
CardExpirationYear	yes	numeric	Numeric year (YY). 2 Digits Max.
CardHolderName	yes	string	CardHolderName. 50 Characters Max.
OrderNumber	yes	string	OrderNumber
ClerkID	yes	string	clerkid for the transaction.
StationID	yes	string	stationid for the transaction.
Comments	yes	string	comment or description for transaction.

Method:CCSettleBatch

returnCCWSResponseStruct
Process Credit Card Settle Batch

Parameters:

Parameters	required	type	description
Login	yes	string	username for service.
Passwd	yes	string	password for service.
Batch	yes	string	TransactionID of successful PREAUTH transaction.

CCWS Response Structure

Field	type	description
status	string	Usually SUCCESS or FAIL.
code	string	Return code from the VaultLine.
auth	string	Authentication code returned from processor.
id	string	Unique transaction id.
cv	string	CVV2 return status.
avs	string	AVS return status.
batch	string	Batch number.
transactionText	string	Message String from processor.

3. Sample Code

Below is provided sample code using Cold Fusion, PHP and .NET.

Note: Due to “intelligent” document formatting please download sample files from the support area of the gateway. These files will be in their native format but compressed (.zip).

Coldfusion Code:

```
<cfparam name="URL.url" default="https://testweb.mydtx.net">

<cfset CCObj = CreateObject("webservice","#URL.url#/CCWS/CCWS.cfc?wsdl")>
<h2>Authorization</h2>
<cftry>
<cfinvoke webservice="#CCObj#" method="authenticate"
        login="#username#" passwd="#password#"
        returnvariable="authResponse">
<cfcatch>
        <cfset authReponse = cfcatch.Message & ":" & cfcatch.Detail>
</cfcatch>
</cftry>
<cfdump var="#authResponse.ToString()#">
<hr>
<h2>Sale</h2>
<cftry>
<cfinvoke webservice="#CCObj#" method="CCSale" returnvariable="saleResponse">
        <cfinvokeargument name="Login" value="#username#">
        <cfinvokeargument name="Passwd" value="#password#">
        <cfinvokeargument name="Amount" value="12.31">
        <cfinvokeargument name="CardNumber" value="4111111111111111">
        <cfinvokeargument name="CardExpirationMonth" value="02">
        <cfinvokeargument name="CardExpirationYear" value="06">
        <cfinvokeargument name="CardHolderName" value="Blah B. Blah">
        <cfinvokeargument name="AVSAddress" value="1234 pittypat lane" >
        <cfinvokeargument name="AVSZip" value="12121" >
        <cfinvokeargument name="CVV2" value="" >
        <cfinvokeargument name="OrderNumber" value="123" >
        <cfinvokeargument name="TaxAmount" value="0">
        <cfinvokeargument name="ClerkID" value="some clerk" >
        <cfinvokeargument name="StationID" value="some ip address" >
        <cfinvokeargument name="Comments" value="This is a comment" >
</cfinvoke>
<cfcatch>
        <cfset saleResponse = cfcatch.Message & ":" & cfcatch.Detail>
</cfcatch>
```

```
</cftry>
<cfset sResponse = dumpResponse(saleResponse)>
<cfdump var="#sResponse#">
<cfset batch = saleResponse.batch>
<cfoutput>#batch#</cfoutput>




<cffunction name="dumpResponse" output="false" returntype="struct">
        <cfargument name="response" required="yes">
        <cfset var rstruct = StructNew()>
        <cfset rstruct.status = response.status>
        <cfset rstruct.code = response.code>
        <cfset rstruct.auth = response.auth>
        <cfset rstruct.id = response.id>
        <cfset rstruct.cv = response.cv>
        <cfset rstruct.avs = response.avs>
        <cfset rstruct.batch = response.batch>
        <cfset rstruct.transactionText = response.transactionText>
        <cfreturn rstruct>
</cffunction>
```

PHP Code:

```
<?php
require_once 'SOAP/Client.php';

$wsdl_url =  'https://testweb.mydtx.net/CCWS/CCWS.cfc?wsdl';
$WSDL     = new SOAP_WSDL($wsdl_url);
$client   = $WSDL->getProxy();
$user = '';
$test = '';
$Amount=12.31;
$cardnumber = "4111111111111111";
$cardexpmonth = "02";
$cardexpyear = "07";
$cardHolderName = "name";
$avsAddr = "";
$avsZip = "";
$cvv2 = "";
$ordno = "1234";
$taxamount = 0;
$clerkid = "me";
$stationid = "123";
$comments = "1234 comments blah";
// don't need to authenticate for things to work as the login and passwd get passed in
anyway.
$result   = $client->authenticate($user,$test);
$result1=$client>CCSale($user,$test,$Amount,$cardnumber,$cardexpmonth,$cardexpye
ar,$cardHolderName,$avsAddr,$avsZip,$cvv2,$ordno,$taxamount,$clerkid,$stationid,$c
omments);
<br>
<hr>
<h2>CCSale</h2>
<?php print_r($result1)?>
<br>
<?php
foreach ($result1 as $obj)
{foreach ($obj as $key => $value)
{echo $key . '=>' . $value . '<br>';}}?> <br>
```

.Net code

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace generic
{
	/// <summary>
	/// Summary description for WebForm1.
	/// </summary>
	public class WebForm1 : System.Web.UI.Page
	{
		protected System.Web.UI.WebControls.Label Label1;
		protected System.Web.UI.HtmlControls.HtmlInputButton
Submit1;

		private void Page_Load(object sender, System.EventArgs e)
		{
			// Put user code to initialize the page here
		}
		protected void Submit1_ServerClick(object sender, EventArgs
e)
		{
			string username = "vitale";
			string passwd = "test";
			double amount = Convert.ToDouble(Request["amount"]);
			string cardNumber = Request["cardNumber"];
			double expMonth = Request["expMonth"];
			double expYear = Request["expYear"];
			string cardHolder = Request["cardHolder"];
			string avsaddr = Request["avsaddr"];
			string avszip = Request["avszip"];
			string cvv2 = Request["cvv2"];
			string ordnum = Request["ordnum"];
			double taxAmount = Convert.ToDouble(0);
			string ClerkID = "aspclerk1";
			string StationID = "aspstation";
			string comments = "asp comments";

			net.mydtx.testweb.CCWS ccwsObj = new
net.mydtx.testweb.CCWS();
			net.mydtx.testweb.CCWSResponseStruct response = new
net.mydtx.testweb.CCWSResponseStruct();

			Label1.Text = "";

			try
			{
				// we really don't need to do this, this is
really here to test
                        // connectivity to the webservice and to make
sure the account
                        // exists.
                        if(!ccwsObj.authenticate(username, passwd))
                              throw new Exception("Could not
authenticate");

                        response = ccwsObj.CCSale(username, passwd,
amount, cardNumber, expMonth, expYear, cardHolder, avsaddr, avszip,
cvv2, ordnum, taxAmount, ClerkID, StationID, comments);

                        Label1.Text += response.status + "<br>";
                        Label1.Text += response.auth + "<br>";
                        Label1.Text += response.code + "<br>";
                        Label1.Text += response.id + "<br>";
                        Label1.Text += response.avs + "<br>";
                        Label1.Text += response.cv + "<br>";
                        Label1.Text += response.transactionText +
"<br>";
                  }
                  catch(Exception ex)
                  {
                        Label1.Text = ex.Message;
                  }
            }


            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                  //
                  // CODEGEN: This call is required by the ASP.NET Web
Form Designer.
                  //
                  InitializeComponent();
                  base.OnInit(e);
            }

            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
                  this.Load += new System.EventHandler(this.Page_Load);

            }
            #endregion
      }
}
```

This uses a proxy object that is created as a Web Reference for the project based on the WSDL URL.

4. Response Codes

1. Transaction Response Result Codes (code)

Result Code	Description
AUTH	transaction authorized
CALL	call processor for authorization
DENY	transaction denied
DUPL	duplicate transaction
PKUP	confiscate card
RETRY	retry transaction
SETUP	setup error
TIMEOUT	transaction not processed in allocated amount of time

2. Address Validation Response Result Codes (AVS)

Result Code	Description
BAD	All checks failed
GOOD	Good avs result
STREET	Street verification failed
UNKNOWN	Result unknown, should treat as good
ZIP	Zip code verification failed

3. CVV2 Response Result Codes (CV)

Result Code	Description
BAD	Verification failed
GOOD	CV verification success
UNKNOWN	Result unknown- should treat as good

4. General User Request Examples

Example: Sale

Request Keys	**Request Values**
Username	vitale
Password	test007
Action	sale
Account	4012888888881
Expdate	0512
Amount	12.00
Street	8320
zip	85284
cv	999
comments	test transaction
ptrannum	123456
Response Keys	**Response Values**
code	AUTH
msoft_code	INT_SUCCESS
phard_code	SUCCESS
auth	123456
avs	GOOD
cv	GOOD
item	1
batch	1
verbiage	APPROVED
timestamp	1121867675
ttid	112

Example: Void

Request Keys	**Request Values**
username	vitale
password	test123
action	void
ttid	112
Response Keys	**Response Values**
code	AUTH
msoft_code	INT_SUCCESS
ttid	112
verbiage	SUCCESS